Vol. 11 No.01 Tahun 2025, pp 58-68
Jurnal Teknologi dan Manajemen Informatika
http:// http://jurnal.unmer.ac.id/index.php/jtmi
P-ISSN: 1693-6604 E-ISSN: 2580-8044

Analisis Perbandingan Algoritma Dijkstra, Haversine, dan Distance Matrix API pada Penentuan Jarak Sekolah di Kota Semarang

Jauharul Umam [1], Khothibul Umam [2], Nur Cahyo Hendro Wibowo [3], Masy Ari Ulinuha[4]

UIN Walisongo Semarang, Indonesia

Article Info

Article History
Received : 29-04-2025
Revised : 04-06-2025
Accepted : 01-07-2025

Keywords
Dijkstra;
Haversine;
Distance Matrix;
OpenStreetMap

Corresponding Author
Khothibul Umam,
UIN Walisongo Semarang,
khothibul_umam@walisongo.ac.id

ABSTRACT

Jarak antara tempat tinggal dan sekolah sering menjadi pertimbangan penting dalam proses pemilihan sekolah, karena berkaitan dengan aksesibilitas, kenyamanan, dan efisiensi waktu tempuh. Terdapat berbagai metode perhitungan jarak yang dapat digunakan dengan keunggulannya masing-masing. Penelitian ini bertujuan untuk membandingkan tiga metode perhitungan jarak, yaitu *Dijkstra* (menggunakan data jaringan jalan dari *OpenStreetMap*), metode *Haversine*, dan *Google Distance Matrix API*. Hasil menunjukkan bahwa *Dijkstra* memberikan estimasi jarak yang lebih realistis dibandingkan metode *Haversine* dengan rata-rata selisih 1,78 km terhadap hasil *Google Distance Matrix API*. Sedangkan *Haversine* memiliki selisih rata-rata sebesar 3.64 km. Penelitian ini menawarkan solusi *offline* berbasis algoritma *Dijkstra* untuk navigasi sekolah di kota besar. Sistem yang dikembangkan menyediakan alternatif estimasi jarak yang efisien dan mandiri untuk keperluan seleksi zonasi dalam PPDB, tanpa ketergantungan pada koneksi internet. Meskipun demikian, sistem ini belum mempertimbangkan faktor dinamis seperti kondisi lalu lintas, serta masih terbatas pada wilayah Kota Semarang dan belum dioptimalkan untuk skenario penggunaan skala besar.

PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi telah mendorong transformasi signifikan dalam bidang transportasi dan navigasi [1]. Seiring dengan meningkatnya mobilitas masyarakat, kebutuhan akan sistem navigasi yang efisien dan akurat menjadi semakin krusial. Salah satu pendekatan populer dalam penentuan rute optimal adalah algoritma *Dijkstra* yang dikenal mampu menghitung jalur terpendek dalam struktur graf berarah [2]. Kota Semarang, sebagai kota besar dengan jaringan jalan yang kompleks, menawarkan tantangan sekaligus peluang untuk penerapan algoritma ini secara lebih kontekstual.

Salah satu konteks penting dari navigasi jarak pendek adalah kebutuhan masyarakat dalam memilih sekolah yang terjangkau secara geografis. Jarak dari rumah ke sekolah menjadi salah satu pertimbangan utama dalam pemilihan sekolah, terutama dalam kaitannya dengan aksesibilitas, kenyamanan transportasi, dan efisiensi waktu tempuh. Aksesibilitas yang baik tidak hanya meningkatkan kenyamanan dan keselamatan siswa, tetapi juga mendukung efektivitas proses belajar mengajar secara keseluruhan. Penelitian oleh Kariematut Thoyyibah dan kawan-kawannya [3] menegaskan bahwa jarak tempuh berperan penting dalam keputusan orang tua dalam memilih institusi pendidikan.

This is an open access article under the CC BY-SA license.
DOI: https://doi.org/10.26905/jtmi.v11i1.15544

Kebijakan Penerimaan Peserta Didik Baru (PPDB) dengan sistem zonasi yang diterapkan di Indonesia semakin menekankan pentingnya jarak antara tempat tinggal calon siswa dan sekolah tujuan. Dalam sistem ini, siswa yang berada dalam zona terdekat memiliki peluang lebih besar untuk diterima di sekolah negeri. Oleh karena itu, penentuan jalur terpendek dan estimasi jarak yang akurat menjadi aspek krusial dalam membantu orang tua dan siswa dalam proses seleksi sekolah, sekaligus menjadi dasar perencanaan wilayah pendidikan yang lebih adil dan merata [4]. Penelitian ini berupaya mengkaji sejauh mana algoritma *Dijkstra* dapat diterapkan secara efektif dan efisien dibandingkan metode *Haversine* dan layanan *Google Distance Matrix API*. Selain itu, penelitian ini juga mengeksplorasi potensi pemanfaatan data *offline* dari *OpenStreetMap* (OSM) sebagai solusi navigasi mandiri dalam mendukung sistem zonasi sekolah di kota besar seperti Semarang.

Pemanfaatan data spasial dari OSM memungkinkan analisis jaringan jalan secara terbuka dan fleksibel [5]. Meskipun demikian, penerapan algoritma *Dijkstra* dalam konteks nyata perlu dibandingkan dengan layanan navigasi berbasis *cloud* seperti *Google Maps API* yang mempertimbangkan variabel dinamis seperti kondisi lalu lintas. Oleh karena itu, penelitian ini mengimplementasikan algoritma *Dijkstra* untuk menghitung jalur terpendek dari lokasi pengguna menuju sekolah-sekolah di Kota Semarang, serta membandingkan hasilnya dengan data dari *Google Maps API* dan metode *Haversine*.

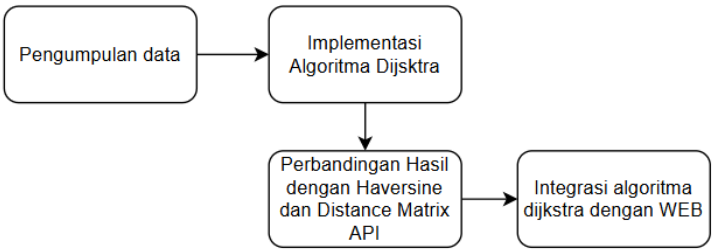
Hasil dari studi ini diharapkan memberikan kontribusi dalam pengembangan sistem navigasi berbasis graf yang dapat beroperasi secara *offline*. Selain itu juga menjadi referensi dalam pemanfaatan data OSM untuk aplikasi transportasi dan GIS, khususnya dalam mendukung sistem zonasi pendidikan seperti PPDB. Penelitian ini bertujuan untuk membandingkan Algoritma *Dijkstra*, metode *Haversine*, dan *Google Distance Matrix API* dalam menentukan jalur terpendek menuju sekolah di Kota Semarang. Masing-masing metode merepresentasikan pendekatan yang berbeda dalam perhitungan jarak *Dijkstra* mengandalkan pemodelan graf yang memungkinkan pemrosesan secara *offline* [6], *Haversine* menggunakan rumus matematis sederhana berbasis koordinat geospasial [7]. Sementara *Google Distance Matrix API* memanfaatkan data dinamis dan infrastruktur *cloud* untuk menyajikan estimasi berbasis kondisi lalu lintas nyata [8]. Perbandingan ini sangat diperlukan untuk mengevaluasi efektivitas dan efisiensi metode navigasi dalam konteks zonasi sekolah berbasis GIS, sekaligus mengkaji potensi penggunaan solusi yang lebih mandiri, terbuka, dan hemat sumber daya.

Beberapa penelitian telah membahas penerapan metode perhitungan jarak dalam sistem informasi geografis, seperti studi oleh Talenta Arta Deva Victoria dan Hermansyah [9] yang berhasil menerapkan Algoritma Dijkstra dalam Pemetaan UMKM Berbasis Android. Penelitian oleh Muhammad Syahputra Novelan [10] juga berhasil menerapkan metode Haversine dalam GIS sebagai penunjuk arah lokasi sekolah terdekat. Namun, kedua studi tersebut masih dilakukan dalam lingkup data dan wilayah yang relatif terbatas. Jika dibandingkan dengan konteks geografis, kota Semarang memiliki kepadatan lokasi dan jaringan jalan yang jauh lebih kompleks. Hal ini memerlukan pendekatan yang mengakomodasi jumlah titik yang lebih besar serta variasi karakteristik wilayah urban. Lebih lanjut, studi oleh Rivaltino Arron dan Angelina Pramana Thenata dalam pengembangan aplikasi cek radius outlet PT.IJS berbasis web menggunakan metode Haversine Formula. Hal ini menunjukkan akurasi rata-rata sebesar 94% dibandingkan dengan pengukuran jarak nyata menggunakan odometer, dengan tingkat kesalahan yang kecil dan masih dapat diterima dalam konteks pemetaan lokasi [11]. Disisi lain, penelitian oleh Nurhamni dan rekan-rekannya yang menerapkan algoritma A* untuk penyortiran lokasi UMKM di Kabupaten Bireun mencatat nilai akurasi tinggi, dengan nilai MAPE (*mean Absolute Percentage Error*) hanya sebesar 3,77% [12]. Hal ini menegaskan keandalan metode pencarian jalur dalam konteks spasial perkotaan maupun non-perkotaan.

Meskipun hasil-hasil serupa pada penelitian terdahulu menunjukkan efektivitas masing-masing metode dalam konteks tertentu, sebagian besar masih terbatas pada jumlah data yang relatif kecil dan area yang sempit. Oleh karena itu, dibutuhkan studi yang menguji kinerja algoritma pencarian jalur dan metode perhitungan jarak pada data yang lebih bervariasi dan

kompleks, terutama dalam skala wilayah urban yang memiliki banyak titik lokasi seperti sekolah. Penelitian ini mencoba menjawab kebutuhan tersebut dengan mengimplementasikan dan membandingkan ketiga pendekatan tersebut dalam konteks geografis Kota Semarang, menggunakan data yang lebih beragam dan jumlah titik lokasi yang lebih banyak. Dengan demikian, pendekatan ini diharapkan dapat memberikan pemahaman yang lebih menyeluruh mengenai kelebihan dan keterbatasan tiap metode dalam situasi nyata yang dinamis. Serta memberikan kontribusi praktis terhadap pengembangan sistem informasi geografis untuk kebutuhan zonasi sekolah.

METODE PENELITIAN



Gambar 1. Metode Penelitian

Gambar 1 menunjukkan alur metode penelitian yang diperoleh secara manual melalui *Google Maps*. Data tersebut kemudian diolah dan digunakan dalam implementasi algoritma *Dijkstra* menggunakan *Python* untuk menghitung jalur terpendek dari satu titik asal ke seluruh sekolah. Hasil perhitungan ini dibandingkan dengan metode *Haversine* dan *Google Distance Matrix API* guna mengevaluasi akurasi dan konsistensi masing-masing pendekatan. Seluruh proses diintegrasikan ke dalam sistem web berbasis Flask yang memungkinkan pengguna mengakses hasil perhitungan secara interaktif melalui antarmuka web.

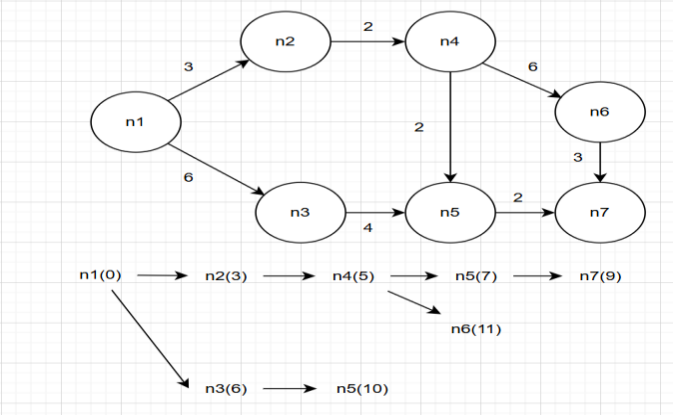
Pengumpulan Data

Penelitian ini diawali dengan proses pengumpulan data spasial jaringan jalan Kota Semarang menggunakan pustaka OSMnx dalam bahasa pemrograman Python. OSMnx memungkinkan ekstraksi data graf dari OSM secara langsung, yang mencakup simpul (*node*) dan ruas jalan (*edge*) berdasarkan wilayah administratif yang telah ditentukan [13]. Data graf ini digunakan sebagai representasi jaringan jalan yang akan dianalisis menggunakan algoritma jalur terpendek. Struktur graf yang dihasilkan bersifat berbobot, dengan bobot berupa jarak antar simpul, yang menjadi basis perhitungan algoritma *Dijkstra*.

Data titik tujuan berupa 368 sekolah yang terdiri dari jenjang SD dan SMP diperoleh dari daftar sekolah yang sudah tersedia sebelumnya. Pemilihan sekolah hanya mencakup dua jenjang tersebut karena Keterbatasan data sekolah jenjang lainnya yang tidak lengkap atau tidak dapat diverifikasi. Selanjutnya, proses penentuan koordinat geografis (*latitude* dan *longitude*) dari masing-masing sekolah dilakukan secara manual melalui *Google Maps*. Setiap sekolah dalam daftar dicari satu per satu, dan koordinatnya dicatat secara presisi, disertai dokumentasi berupa foto bangunan sekolah untuk keperluan visualisasi pada antarmuka web.

Implementasi Algoritma Dijsktra

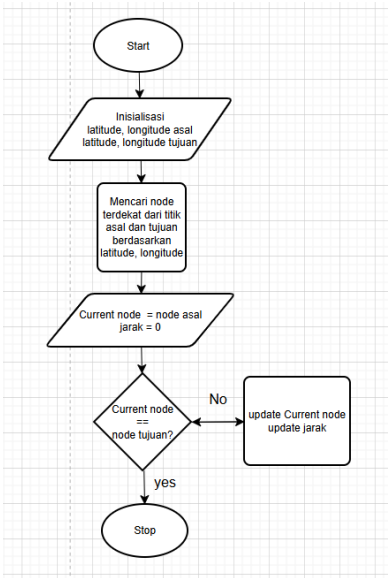
Algoritma *Dijkstra* merupakan salah satu algoritma graf klasik yang digunakan untuk menyelesaikan masalah *single-source shortest path*, yaitu mencari jarak terpendek dari satu titik asal ke seluruh titik lainnya dalam sebuah graf berbobot non-negatif. Prinsip kerja algoritma ini didasarkan pada pendekatan *greedy*, yaitu secara iteratif memilih simpul dengan jarak terkecil yang belum dikunjungi, kemudian memperbarui jarak minimum ke simpul-simpul tetangganya hingga seluruh simpul terjangkau [14].



Gambar 2. Contoh *Graph* Berarah

Gambar 2 merupakan contoh bagaimana algoritma *dijkstra* dijalankan. Diagram bagian atas menggambarkan ilustrasi graf berarah. Sedangkan diagram di bawah menunjukkan langkah demi langkah menghitung jarak dari node asal (n1) sampai node tujuan (n7). Langkah pertama dimulai dari simpul n1, yang memiliki jarak 0. Dari simpul ini, dilakukan pembaruan jarak ke simpul tetangganya, yaitu n2 (jarak 3) dan n3 (jarak 6). Kemudian, simpul dengan jarak terkecil berikutnya, n2, dipilih untuk diproses, dan dilakukan pembaruan jarak ke simpul n4 (total jarak 5). Selanjutnya, simpul n4 diproses dan memperbarui jarak ke simpul n5 (jarak 7) dan n6 (jarak 11). Setelah itu, simpul n3 diproses, namun jarak ke simpul n5 (total 10) lebih besar dibanding jarak sebelumnya (7), sehingga tidak diperbarui. Proses dilanjutkan ke simpul n5, yang memperbarui jarak ke simpul n7 menjadi 9. Simpul-simpul terakhir yang diproses adalah n7 dan n6, namun tidak ada lagi pembaruan jarak.

Hasil akhir dari proses ini menunjukkan jarak terpendek dari simpul n1 ke simpul n7 adalah n1, n2, n4, n5, dan n7 dengan total jarak bobot 9. Proses ini menggambarkan efisiensi algoritma *Dijkstra* dalam menentukan jalur optimal dari simpul sumber ke seluruh simpul lainnya dalam graf berbobot. Dalam penelitian ini, algoritma *Dijkstra* diimplementasikan menggunakan bahasa pemrograman *Python*. Bahasa ini dipilih karena memiliki sintaks yang sederhana, mendukung pengolahan data spasial, serta fleksibel dalam membangun logika algoritmik [15].



Gambar 3. Implementasi Algoritma Dijkstra

Proses dimulai dengan menginisialisasi koordinat titik asal (lokasi pengguna) dan titik tujuan (lokasi sekolah). Karena jaringan jalan direpresentasikan sebagai graf, koordinat tersebut

kemudian dicocokkan ke simpul (*node*) terdekat dalam graf. Setelah simpul asal dan tujuan ditemukan, algoritma *Dijkstra* dijalankan untuk menghitung jalur terpendek.

Perhitungan dilakukan secara iteratif yang dimulai dari simpul asal. Jika simpul tersebut belum mencapai tujuan, maka jarak ke simpul-simpul tetangga diperbarui, dan simpul dengan jarak terkecil dipilih sebagai simpul berikutnya. Proses ini berlanjut hingga simpul tujuan tercapai, menghasilkan jalur terpendek dari asal ke tujuan. Dalam penerapannya algoritma ini dapat secara dinamis menyesuaikan rute berdasarkan perubahan kepadatan jalan untuk memberikan waktu tempuh yang optimal.

3. Perbandingan dengan Distance Matrix API dan Haversine

Sebelum algoritma *Dijkstra* diintegrasikan ke dalam sistem berbasis web, dilakukan proses evaluasi hasil perhitungan jarak dengan membandingkannya terhadap dua pendekatan lain, yaitu *Google Maps Distance Matrix API* dan rumus *Haversine*. *Distance Matrix API* dari *Google* memungkinkan perhitungan jarak berdasarkan data jaringan jalan terkini yang mempertimbangkan kondisi dinamis seperti lalu lintas [16] Namun API ini memiliki Keterbatasan teknis, yaitu hanya dapat memproses maksimal 25 titik tujuan per permintaan (*batch*), sehingga diperlukan pengelompokan data secara bertahap untuk mencocokkan seluruh pasangan asal-tujuan secara efisien. Sementara itu, metode *Haversine* menghitung jarak lintas lurus antar dua titik koordinat geografis berdasarkan rumus geodesi bola bumi [17]. Rumus *Haversine* dinyatakan sebagai berikut [18]:

$$d = 2r\ sin^{-1}\left(\sqrt{sin^2\left(\frac{\varphi 2 - \varphi 1}{2}\right) + cos(\varphi 1)cos(\varphi 2)sin^2\left(\frac{\psi 2 - \psi 1}{2}\right)}\right)$$

Dimana:

φ2, φ1	= latitude tujuan, latitude asal
ψ2, ψ1	= longitude tujuan, longitude asal
R	= Konstanta jari-jari bumi (± 6,371 km)
d	= jarak antara dua titik

Pengujian dilakukan dari satu titik asal ke seluruh titik tujuan (368 sekolah), sehingga menghasilkan data jarak dari satu titik ke banyak titik menggunakan ketiga metode. Pemilihan satu titik asal dilakukan untuk menjaga konsistensi dalam proses perbandingan hasil antar metode dalam wilayah administratif kota Semarang dan dianggap cukup untuk memberikan ilustrasi awal terhadap performa masing-masing metode. Pendekatan ini juga memungkinkan fokus pada analisis perbedaan nilai jarak yang dihasilkan tanpa adanya variabel tambahan dari perbedaan lokasi titik asal.

Integrasi algoritma dijkstra dengan web GIS

Setelah dievaluasi, algoritma Dijkstra diintegrasikan ke dalam aplikasi web berbasis Flask untuk mengelola *backend* dan *routing*. Antarmuka web dirancang agar dapat mendeteksi lokasi pengguna melalui *browser geolocation* API (dengan izin akses), lalu menghitung jarak terpendek ke setiap sekolah menggunakan algoritma *Dijkstra*. Hasil perhitungan ditampilkan dalam daftar jarak (kilometer) secara langsung, memungkinkan pengguna memperoleh informasi jarak ke sekolah-sekolah di Kota Semarang secara efisien melalui web.

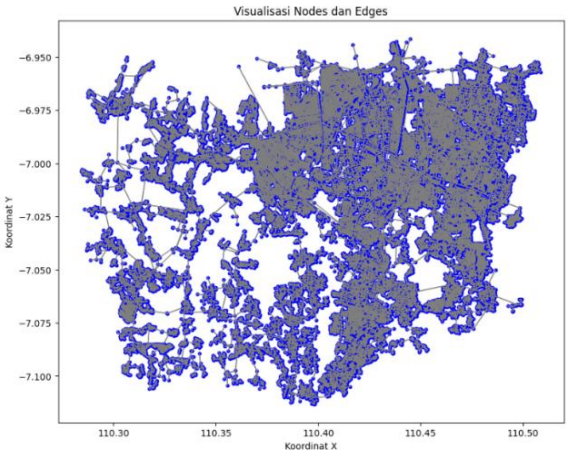
HASIL DAN PEMBAHASAN

Hasil pengumpulan dan Pengolahan Data Jaringan Jalan

Data jaringan jalan yang digunakan dalam penelitian ini diperoleh dari OSM, sebuah platform pemetaan global berbasis komunitas yang menyediakan data geospasial secara terbuka [19]. Untuk keperluan penelitian ini, data OSM diolah menggunakan pustaka Python OSMnx. OSMnx memungkinkan ekstraksi data jaringan jalan dari OSM dalam bentuk graf, yaitu struktur data yang terdiri atas simpul (*node*) dan sisi (*edge*). Representasi graf ini

memungkinkan dilakukannya analisis perhitungan jalur terpendek dengan algoritma *Dijkstra*. Struktur graf yang diperoleh terdiri dari:

- ***Node***: Merepresentasikan titik persimpangan atau ujung jalan. Setiap node memiliki atribut seperti ID unik, koordinat lintang dan bujur, serta jumlah jalan yang terhubung.
- ***Edge***: Menghubungkan dua simpul dan menyimpan informasi seperti ID jalan, jenis jalan (highway), jumlah jalur (lanes), panjang jalan (length), arah jalan (oneway), dan geometri jalan.
- ***Atribut Graph***: Metadata umum yang disimpan mencakup waktu pembuatan graf, perangkat lunak yang digunakan, sistem koordinat (WGS 84 / EPSG:4326), dan status penyederhanaan graf.

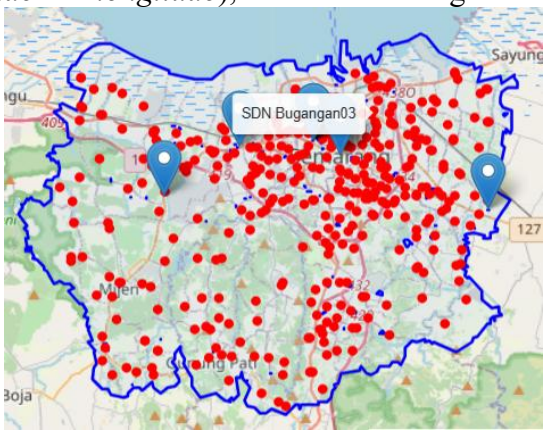


Gambar 4. Visualisasi Node dan Edge di kota Semarang

Gambar 4 merupakan visualisasi graf jaringan jalan kota Semarang, yang dibangun berdasarkan data spasial dari *OpenStreetMap* menggunakan pustaka OSMnx. Graf ini merepresentasikan *node* sebagai titik persimpangan atau ujung jalan, serta *edge* sebagai ruas jalan yang menghubungkan antar simpul. Statistik dari hasil ekstraksi menunjukkan total 48.692 simpul dan 124.915 sisi. Visualisasi graf ini menggambarkan jaringan jalan kota Semarang dalam bentuk jaringan yang kompleks, memungkinkan pemodelan jalur secara detail serta penerapan algoritma graf untuk analisis rute.

Hasil Pengumpulan Data Sekolah Kota Semarang

Data sekolah diperoleh melalui pencatatan manual berdasarkan informasi dari *Google Maps*, mencakup 368 titik lokasi sekolah (SD dan SMP). Data ini terdiri dari nama sekolah, koordinat geografis (*latitude* dan *longitude*), serta foto masing-masing sekolah.



Gambar 5. Distribusi Sekolah-sekolah berupa SD dan SMP di Kota Semarang

Gambar 5 memberikan gambaran distribusi spasial sekolah-sekolah di wilayah Kota Semarang, yang divisualisasikan dengan penanda berupa titik merah. Adapun garis biru menunjukkan batas administratif resmi kota Semarang. Seluruh titik sekolah yang

teridentifikasi pada peta ini akan digunakan sebagai titik tujuan dalam perhitungan jarak dari posisi geografis pengguna.

Implementasi Algoritma *Dijkstra*

Implementasi algoritma *Dijkstra* dimulai dengan menentukan lokasi awal (lokasi pengguna) dan mencari *node* terdekat pada lokasi awal dan tujuan (sekolah-sekolah) dalam graf jaringan jalan menggunakan struktur data *kd_tree*. Struktur ini mempercepat proses pencarian *node* terdekat dalam ruang dua dimensi. Setelah *node* awal dan *node* tujuan (sekolah) ditentukan, algoritma *Dijkstra* menghitung bobot terpendek antar *node* dalam graf dengan cara sebagai berikut:

- Menginisialisasi bobot semua *node* sebagai tak hingga, kecuali *node* awal yang memiliki bobot nol.
- Menggunakan prinsip *greedy*, algoritma memilih *node* dengan bobot terkecil dan memperbarui bobot *node* tetangga jika ditemukan jalur yang lebih pendek.
- Proses ini dilakukan iteratif hingga ditemukan jalur terpendek dari titik awal ke titik tujuan.

Hasil dari algoritma ini berupa jalur optimal dari pengguna ke sekolah terdekat berdasarkan struktur graf jaringan jalan kota Semarang.

Perbandingan dengan *Google Maps Distance Matrix API* dan *Haversine*



Gambar 6. Visualisasi Hasil Perbandingan Tiga Metode

Gambar 6 memperlihatkan perbandingan visual rute dan jarak yang dihitung antara dua titik menggunakan tiga metode berbeda, yaitu *Google Maps API* (warna merah), algoritma *Dijkstra* (warna biru), dan metode *Haversine* (warna hijau). Perbedaan warna pada peta memudahkan pemahaman variasi jalur dan jarak yang dihasilkan oleh masing-masing metode. Berdasarkan hasil perhitungan jarak antara dua titik menggunakan tiga metode yang berbeda, diperoleh nilai jarak sebagai berikut: metode *Dijkstra* menghasilkan jarak sejauh 10 km, *Google Maps API* menghasilkan jarak 11 km, dan metode *Haversine* menghasilkan jarak 8,6 km. Ketiga metode tersebut menunjukkan hasil yang bervariasi karena pendekatan dan sumber data yang digunakan dalam perhitungannya berbeda.

Metode *Haversine* menghitung jarak garis lurus antar koordinat berdasarkan asumsi bumi sebagai bola sempurna. Sehingga hasilnya cenderung lebih pendek karena mengabaikan struktur jalan dan kondisi geografis. *Google Distance Matrix API* lebih mendekati kenyataan dengan mempertimbangkan jalur utama, struktur jalan, dan lalu lintas *real-time*. Sementara itu, algoritma *Dijkstra* menggunakan data jaringan jalan dari *OpenStreetMap* untuk menghitung rute terpendek secara deterministik, mencerminkan struktur jalan aktual meskipun tanpa mempertimbangkan kondisi lalu lintas.

Dari sisi waktu eksekusi (dalam menghitung jarak satu titik ke 368 sekolah), terdapat perbedaan yang cukup signifikan di antara ketiga metode. Metode *Haversine* merupakan yang tercepat dengan waktu eksekusi hanya 0,0114 detik karena perhitungannya sangat sederhana secara matematis dan tidak membutuhkan pemrosesan data spasial yang kompleks. Metode *Dijkstra* membutuhkan waktu sekitar 2,1899 detik karena melibatkan pemrosesan graf jaringan jalan yang cukup besar, meskipun berjalan secara lokal dan *offline*. *Google Maps API* (dengan

25 data per-batch) merupakan yang paling lambat dengan waktu eksekusi 2,98 detik karena membutuhkan permintaan ke server eksternal serta proses parsing data dari respons API.

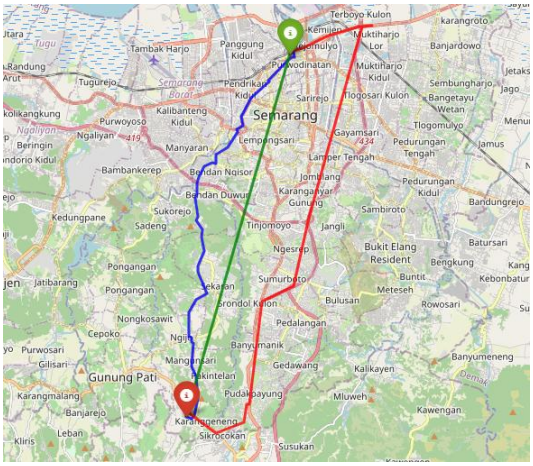
Setiap metode memiliki kelebihan dan kekurangannya masing-masing. Algoritma *Dijkstra* memiliki keunggulan karena dapat berjalan secara *offline*, gratis, serta relatif cepat, namun memerlukan pemutakhiran data jaringan jalan secara manual jika terjadi perubahan graf dan tidak mampu memperhitungkan kondisi lalu lintas. *Google Distance Matrix API* unggul dalam menyajikan informasi secara *real-time* dan sangat relevan dengan kondisi aktual di lapangan, namun penggunaannya dibatasi oleh biaya dan ketergantungan pada koneksi internet. Sementara itu, metode *Haversine* sangat cocok untuk estimasi awal atau perhitungan jarak antar koordinat secara cepat, namun tidak dapat digunakan untuk penentuan rute nyata karena tidak mempertimbangkan bentuk dan struktur jaringan jalan.

Tabel 1. Ringkasan Perbandingan Hasil Dijkstra, Google Maps API, dan Haversine

Statistic	Latitude	Longitude	Distance_api	Distance_ haversine	Distance_ dijkstra	Abs (dijkstra-api)	Abs (haversine-api)
count	368.000000	368.000000	368.000000	368.000000	368.000000	368.000000	368.000000
mean	-7.012285	110.406998	11.334758	7.699348	9.566685	1.776970	3.635410
std	0.039144	6.810578	6.180758	4.520570	5.457653	1.928434	2.643462
min	-7.110116	110.287649	0.480000	0.410000	1.000000	0.020000	0.000000
25% (Q1)	-7.039823	110.376118	5.942750	4.052500	5.377500	0.260000	1.549500
50% (Median)	-7.012330	110.416355	9.319500	6.655000	9.320000	0.820000	2.616000
75% (Q3)	-6.981939	110.431820	17.364500	11.390000	13.570000	2.637000	5.622250
max	-6.950247	110.450961	27.478000	18.790000	23.590000	8.598000	11.648000

Tabel 1 menunjukkan Analisis statistik deskriptif dari selisih jarak hasil perhitungan antara metode *Dijkstra*, *Google Maps API*, dan *Haversine* yang memberikan hasil nilai minimum, maksimum, rata-rata, dan kuartil dari selisih jarak yang diperoleh dari pengujian satu titik asal pengguna terhadap 368 lokasi sekolah. Statistik ini memberikan gambaran awal mengenai sebaran dan variasi akurasi masing-masing metode. Data jarak *Distance Matrix API* diakses pada hari Kamis, 29 Mei 2025. Perbandingan akurasi menunjukkan bahwa rata-rata selisih jarak antara Dijkstra dan Google Maps API sebesar 1,78 km. Sedangkan Haversine mencapai 3,64 km.

Perbedaan ini muncul karena *Dijkstra* dan API mempertimbangkan jaringan jalan nyata, sementara *Haversine* hanya menghitung jarak lurus antar titik. Dalam beberapa kasus ekstrem, selisih *Haversine* dan API mencapai 11,65 km, seperti pada Gambar 7, *Haversine* menghasilkan rute lurus yang melintasi pegunungan, sementara API memilih rute memutar melalui jalan tol. Hal ini menegaskan keterbatasan *Haversine* dalam mencerminkan kondisi geografis dan struktur jalan sebenarnya.



Gambar 7. Visualisasi Hasil Perbandingan Jarak

Analisis dan Interpretasi Hasil

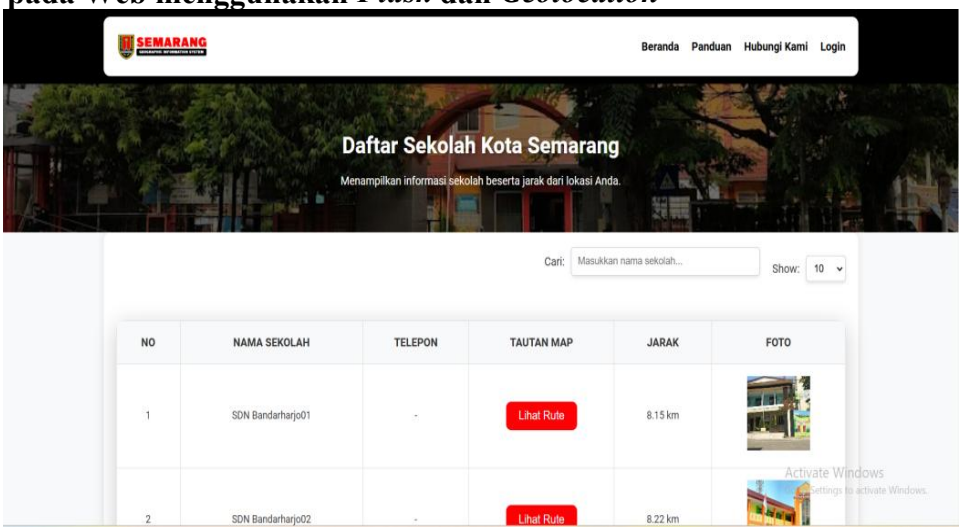
Metode *Haversine* sangat cepat dan efisien dalam hal komputasi, namun tidak dapat digunakan untuk navigasi karena hanya mempertimbangkan jarak lurus [20]. Sebaliknya, *Google Distance Matrix API* sangat akurat dan *real-time*, tetapi bergantung pada layanan pihak ketiga dan dapat menimbulkan biaya. Dari hasil perhitungan dan visualisasi, algoritma *Dijkstra* menunjukkan hasil yang cukup mendekati jalur aktual dari *Google Distance Matrix API*, dengan waktu eksekusi yang relatif cepat dan hasil jarak rute yang lebih efisien dibandingkan API. Untuk menguji apakah terdapat perbedaan signifikan antara jarak yang dihitung oleh algoritma *Dijkstra* dan *Google Distance Matrix API*, dilakukan analisis *paired t-test* terhadap 368 jarak dengan hasil sebagai berikut:

- *t-statistic*: -17.5138
- *p-value*; <0.000001
- Rata-rata selisih: -1.7681 km
- 95% *confidence Interval*: [-1.9659, -1.5702]

Berdasarkan nilai p yang sangat kecil, dapat disimpulkan bahwa terdapat perbedaan signifikan secara statistik antara hasil *Dijkstra* dan *Google Distance Matrix API*. Selisih negatif menunjukkan bahwa *Dijkstra* secara konsisten menghasilkan jarak yang lebih pendek. Hal ini terjadi karena *Dijkstra* menghitung jalur terpendek berdasarkan struktur graf jalan, tanpa mempertimbangkan kecepatan atau kondisi lalu lintas. Sedangkan *Google API* sering memilih rute lebih jauh namun lebih cepat seperti jalan tol.

Dengan karakteristik tersebut, *Dijkstra* lebih sesuai untuk aplikasi yang membutuhkan akurasi pengukuran jarak spasial. Seperti sistem zonasi sekolah (PPDB), panjang lintasan lebih relevan dari pada waktu tempuh. Namun, sistem memiliki keterbatasan cakupan spasial karena graf hanya mencakup wilayah kota Semarang. Jika digunakan oleh pengguna di luar wilayah tersebut, akurasi jarak menjadi terganggu karena titik awal berada di luar graf. Selain itu, pemrosesan rute secara lokal dapat membebani sistem saat diakses secara bersamaan. Sehingga diperlukan strategi pengelolaan beban agar sistem tetap responsif.

Integrasi pada Web menggunakan *Flask* dan *Geolocation*



Gambar 8. Hasil Integrasi dengan Algoritma *Dijsktra*

Gambar 8 menunjukkan hasil integrasi web dengan algoritma *Djikstra.* Agar algoritma *Dijkstra* dapat digunakan dalam aplikasi web, sistem diintegrasikan dengan *Flask* sebagai *backend* untuk menangani permintaan data dan komunikasi dengan *frontend* [21]. Di sisi *frontend*, koordinat pengguna diperoleh melalui *navigator* dan *geolocation* yang meminta izin akses lokasi dari *browser* [22]. Setelah mendapatkan koordinat *latitude* dan *longitude*, data dikirim ke server untuk perhitungan jalur terpendek. Hasil perhitungan kemudian dikembalikan ke *frontend* dan ditampilkan pada antar muka pengguna.

SIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa algoritma *Dijkstra* menghasilkan estimasi jarak yang lebih akurat dibandingkan metode *Haversine.* Dengan rata-rata selisih sebesar 1,78 km terhadap *Google Distance Matrix API*, sementara *Haversine* mencapai 3,64 km. Hasil ini menegaskan bahwa *Dijkstra* lebih sesuai untuk sistem navigasi sekolah di kawasan perkotaan seperti kota Semarang. Sistem yang dibangun mampu berjalan secara *offline* dengan memanfaatkan data dari *OpenStreetMap*. Sehingga menjadi alternatif hemat biaya dan independen dari layanan pihak ketiga. Temuan ini berpotensi meningkatkan transparansi dan efisiensi dalam kebijakan zonasi PPDB.

Pada penelitian berikutnya dapat dikembangkan melalui penerapan algoritma *Dijkstra* pada graf dinamis untuk menghasilkan jarak yang lebih adaptif terhadap perubahan kondisi jaringan jalan. Pendekatan *caching* juga dapat dimanfaatkan guna mengurangi redundansi perhitungan dan meningkatkan efisiensi pada rute yang sering diakses. Selain itu, diperlukan pengujian pada skenario multi-pengguna serta eksplorasi optimalisasi komputasi melalui pemrosesan paralel. Cakupan sistem pun dapat diperluas ke wilayah lain di luar Kota Semarang agar implementasinya menjadi lebih luas dan kontekstual.

DAFTAR PUSTAKA

[1] H. O. Abdullahi, I. H. Mohamud, A. F. Ali, A. A. Hassan, and A. Kafi, "The Transformative Impact of Information and Communication Technology on Transportation Services: A Systematic Literature Review," *Int. J. Transp. Dev. Integr.*, vol. 8, no. 3, pp. 455–460, 2024, doi: https://doi.org/10.18280/ijtdi.080309

[2] F. Kuncoro, I. A. Zulkarnain, and G. A. Buntoro, "Application of Dijkstra's Algorithm in Determining the Shortest Route to Mrican Landfill," *Antivirus J. Ilm. Tek. Inform.*, vol. 18, no. 2, pp. 200–211, 2024, doi: https://doi.org/10.35457/antivirus.v18i2.3785

[3] K. Thoyyibah, D. R. Adhimah, and R. Dewi Lukitasari, "Analisis Faktor Pertimbangan Orang Tua Dalam Memilihkan Sekolah," *Pros. Semin. Nas. UNIMUS*, vol. 5, pp. 702–725, 2022.
Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia Nomor 47/M/2023 .../.../... tentang Pedoman Pelaksanaan Peraturan Menteri Pendidikan dan Kebudayaan Nomor 1 Tahun 2021 tentang Penerimaan Peserta Did," Jakarta, 2023.

[5] I. Supriadi, N. A. P. Siregar, M. R. B. Alfyansyah, and A. R. Putra, "Pengembangan Sistem Menggunakan OpenStreetMaps Api dengan Media Leafletjs Berdasarkan Kategori Gedung Milik Pemerintahaan di Daerah Kota Bandung," *J. Ilm. Ilk. - Ilmu Komput. Inform.*, vol. 8, no. 1, pp. 87–98, 2025, doi: https://doi.org/10.47324/ilkominfo.v8i1.317

[6] R. I. V. Pasaribu and S. Yurinanda, "Optimasi Rute Terpendek Pemeliharaan Lisrik Dengan Algoritma Dijkstra di PLN UIP Sumbagsel," *JISTech (Journal Islam. Sci. Technol.*, vol. 9, no. 2, pp. 240-248, 2024, doi: http://dx.doi.org/10.30829/jistech.v9i2.22446

[7] R. Herwanto, F. Susanto, R. Dwi, M. H. Prayoga, R. M. Dinata, and W. Wamiliana, "Haversine Geo-Spasial Data Android Model Untuk Optimasi Rute Kebersihan Lingkungan Terdekat," *Jurnal Pepadun*, vol. 5, no. 1, pp. 83–92, 2024, doi: https://doi.org/10.23960/pepadun.v5i1.201

[8] P. Chavan *et al.*, "Leveraging Real-Time Data: A Location-based Ambulance Booking andnd Tracking System With Geofencing," *J. Integr. Sci. Technol.*, vol. 13, no. 2, 2024, doi: https://doi.org/10.62110/sciencein.jist.2025.v13.1039

[9] T. A. D. Victoria and Hermansyah, "Penerapan Algoritma Dijkstra dalam Pemetaan UMKM Berbasis Android," *Bull. Comput. Sci. Res.*, vol. 3, no. 6, pp. 420–426, 2023, doi: https://doi.org/10.47065/bulletincsr.v3i6.276

[10] M. S. Novelan, "Penerapan GIS (Geographic Information System) Penunjuk Arah Lokasi Sekolah Terdekat Menggunakan Metode Haversine," *SATESI J. Sains Teknol. dan Sist. Inf.*, vol. 2, no. 1, pp. 1–5, 2022, doi: https://doi.org/10.54259/satesi.v2i1.623

[11] R. Arron and A. P. Thenata, "Perancangan Aplikasi Cek Radius Outlet PT. IJS Berbasis Web Menggunakan Metode Haversine Formula," *Zo. J. Sist. Inf.*, vol. 6, no. 2, pp. 437–448, 2024, doi: https://doi.org/10.31849/zn.v6i2.14349

[12] Nurhamni, "Geographic Information System (GIS) Uses A* Algorithm for Sorting Nearest Umkm Locations," *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 2, pp. 529-538, 2025, doi:

https://doi.org/10.23960/jitet.v13i2.6256
[13] G. Boeing, "Modeling and Analyzing Urban Networks and Amenities with OSMnx," *Geographical Analysis: An International Journal of Theoretical Geography,* vol. 57, no. 4, pp. 1–16, 2024, doi: https://doi.org/10.1111/gean.70009
[14] E. C. Rufus, R. R. Riyadi, D. N. Hasibuan, E. Christian, and V. H. Pranatawijaya, "Penerapan Algoritma Dijkstra dalam Menentukan Rute Terpendek untuk Jasa Pengiriman Barang di Palangka Raya," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 8, no. 3, pp. 3387–3391, 2024, doi: https://doi.org/10.36040/jati.v8i3.9683
[15] J. Iskandar, H. Suhendar, and B. D. Pamungkas, "Analisis Strategi Algoritma Sorting Menggunakan Metode Komparatif pada Bahasa Pemrograman Java dengan Python," *G-Tech J. Teknol. Terap.*, vol. 8, no. 1, pp. 104–113, 2023, doi: https://doi.org/10.33379/gtech.v8i1.3556
[16] A. S. Shibghatullah, A. Jalil, M. H. A. Wahab, J. N. P. Soon, K. Subaramaniam, and T. Eldabi, "Vehicle Tracking Application Based on Real Time Traffic," *Int. J. Electr. Electron. Eng. Telecommun.*, pp. 67–73, 2022, doi: https://doi.org/10.18178/ijeetc.11.1.67-73
[17] R. Palupi, D. A. Yulianna, and S. S. Winarsih, "Analisa Perbandingan Rumus Haversine Dan Rumus Euclidean Berbasis Sistem Informasi Geografis Menggunakan Metode Independent Sample t-Test," *JITU: J. Inform. Technol. Commun.*, vol. 5, no. 1, pp. 40–47, 2021, doi: https://doi.org/10.36596/jitu.v5i1.494
[18] Y. S. Purwanto, M. Farid Rifai, H. Jatnika, and T. M. T. Purba, "Android-Based Community Security And Order Monitoring Application Using Haversine Formula," *Jurnal Teknik Informatika dan Sistem Informasi,* vol. 10, no. 1, pp. 961–971, 2023.
[19] A. Y. Grinberger, M. Minghini, L. Juhász, G. Yeboah, and P. Mooney, "OSM Science—The Academic Study of the OpenStreetMap Project, Data, Contributors, Community, and Applications," *ISPRS Int. J. Geo-Information*, vol. 11, no. 4, pp. 230, 2022, doi: https://doi.org/10.3390/ijgi11040230
[20] A. Riswandi, I. Zufria, and D. Irawan, "Sistem Informasi Geografis Untuk Monitoring Menara Telekomunikasi Menggunakan Metode Haversine Berbasis Android," *J. Ilm. Bin. STMIK Bina Nusant. Jaya Lubuklinggau*, vol. 5, no. 1, pp. 15–21, 2023, doi: https://doi.org/10.52303/jb.v5i1.89
[21] M. A. Fadilla, M. F. Sholahuddin, and T. Sutabri, "Pengembangan Sistem Klasifikasi Diagnosa Medis Menggunakan Progressive Web Application Terintegrasi Machine Learning," *J. Syntax Admiration*, vol. 5, no. 12, pp. 5488–5503, 2024, doi: https://doi.org/10.46799/jsa.v5i12.1906
[22] A. Kurnianti, H. Setyawan, A. P. Santika, and R. Prakosa, "Pengembangan Si Geografis Lokasi TKA dan TPA di Bawah Naungan Badko Bantul Berbasis Android," *DedikasiMU J. Community Serv.*, vol. 4, no. 4, pp. 400-409, 2022, doi: https://doi.org/10.30587/dedikasimu.v4i4.4644